1 Objectives

The purposes of this lecture is to demonstrate how functions themselves can represent arbitrary data structures.

- Know the three constructs of λ-calculus.
- Know the competitive advantage of λ-calculus.
- Know how to use functions to represent integers.
  - Know how to write addition, multiplication, and exponentiation.
- Know how to use functions to represent arbitrary types.
  - booleans
  - pairs
  - lists

2 Examples

We can implement this very easily in OCaml.
$\lambda x.x =$ `fun x -> x`

Variables $\dfrac{}{x \Downarrow x}$

Functions $\dfrac{}{\lambda x.M \Downarrow \lambda x.M}$

Application $\dfrac{M \Downarrow \lambda x.P \quad N \Downarrow V}{M\ N \Downarrow [V/x]P}$

`f0 = fun f x -> x`
`f1 = fun f x -> f x`
`f2 = fun f x -> f (f x)`
`f3 = fun f x -> f (f (f x))`

```
# let fAdd m n f x = m f (n f x);;
val fAdd : ('a -> 'b -> 'c) ->
           ('a -> 'd -> 'b) -> 'a -> 'd -> 'c = <fun>
# fShow (fAdd f3 f3);;
- : int = 6
# let fMul m n f x = m (n f) x;;
val fMul : ('a -> 'b -> 'c) ->
           ('d -> 'a) -> 'd -> 'b -> 'c = <fun>
# fShow (fMul f3 f3);;
- : int = 9
```

3 Problems

Try the following problems. In a few minutes the instructor will go over the solutions. Feel free to work with the person next to you!

1. (126) Write a function `church n` which returns the Church Numeral for n. (Use recursion.)
2. (127) Write the church function for exponentiation. (Hint: the solution is shorter than multiplication.) Show a sample run for 3^2.
3. (128) How would you represent a tree using functions? (There are many ways to do this.)